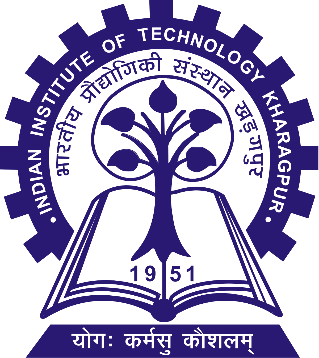
**NSSC 2021**

**Data Analytics**

**Team name : Eccentric**

**Identification and Analysis of Transiting Exoplanet Signals**

**TEAM MEMBERS:**

* **Kshitiz Kumawat**
* **Patel Manthan Shaileshkumar**
* **Meghashrita Das**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Topic | Page. No. |
| 1 | Environment Setup : Platform, Programming language & Libraries | 1 |
| 2 | Task 1 : Plotting and analysis of noisy transit signals | 2 |
| 3 | Task 2 : Transit signal detection and analysis | 5 |
| 4 | Task 3 : Estimation of stellar parameters using Asteroseismology | 9 |
| 5 | Task 4 : Estimation of planet parameters from transit data | 16 |
| 6 | Conclusion | 18 |
| 7 | References | 18 |

**Environment Setup**

**Platform : Google Colab**

**You can find all the codes for this report in below notebook**

[**https://colab.research.google.com/drive/1bSbDjnKsMhIdSa27Wp2-jM7i3znds6Fz?usp=sharing**](https://colab.research.google.com/drive/1bSbDjnKsMhIdSa27Wp2-jM7i3znds6Fz?usp=sharing)

**Programming Language : Python**

**Libraries : Lightkurve and NumPy**

**Setup** –

# install lightkurve library

!pip install lightkurve

# import all the useful library for data analytics

import lightkurve as lk   # for handling lightcurve data

import numpy as np        # for mathematical functions

**Note**

* **All Code snippets are contained in boxes.**
* **All the results are highlighted with Yellow color.**
* **All the questions are attempted in sequential order**

**1. Plotting and Analysis of noisy transit signals**

**(A) Search and download all the light curve data of the star Kepler-17**

**Approach –**

* We will use the [lightkurve](https://docs.lightkurve.org/) library to download the lightcurve data for the star Kepler-17.
* For this we will require either KIC ID = 10619192 or TIC ID = 273874849

**Code** –

# searches all file with KIC ID = 10619192

pixelfile = lk.search\_targetpixelfile("KIC 10619192")

# downloads all the 46 files

pixelfile.download\_all()

**Explanation** –

Here we used KIC ID = 10619192 to extract the pixel file data of Kepler-17. When we hit the download command all the 46 files got downloaded in our system. We will convert these pixel files to lightcurve later.

**(B) Obtain the best-suited light curve and plot it**

**Approach –**

* We have 46 files and we don’t know which will give best plot.
* So, we will iterate through all the 46 files and plot their lightcurves.
* We will choose the one which has less trends and noises. That would be the best

**Code** –

# iterate through all 46 files

for i in range(46):

  # search and download the pixelfile

  pixelfile = search\_targetpixelfile("KIC 10619192")[i].download()

  # convert the pixelfile to lightcurve by following command

  lc = pixelfile.to\_lightcurve(aperture\_mask='all')

  # plot the lightcurve

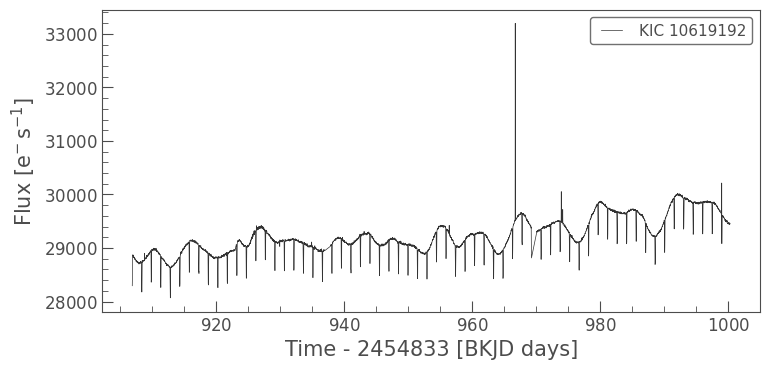
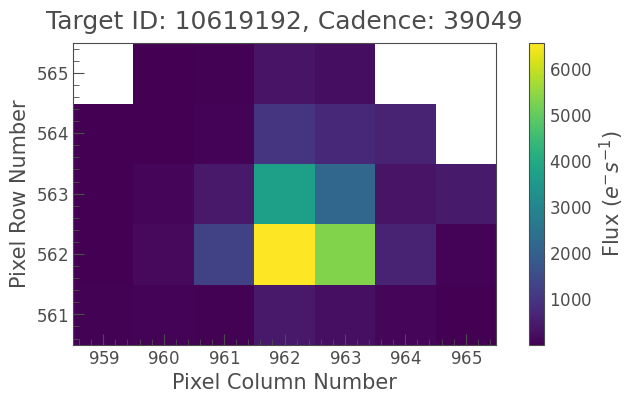
  lc.plot()

**Explanation** –

This will help us to figure out the best plot among all and it can improve our calculations.

**Results** –

We got the following as the best plot. We will visualize its pixelfile and lightcurve.



**(C) Modify the above-plotted graph by normalizing it.**

**Approach –**

* For normalization, we will use the “normalize()” function in lightkurve library
* Then we will plot the curve by “plot()” function.

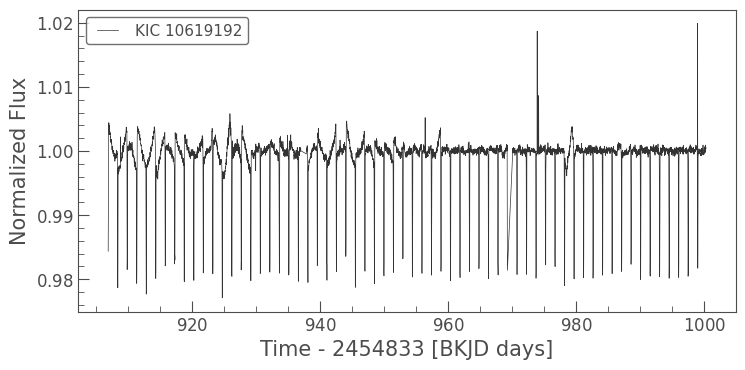
**Code** –

# normalizes the lightcurve

lc\_normalized = lc.normalize()

# plot the normalized curve

lc\_normalized.plot()

****

**(D) Estimate the following information about the light curve –**

**1. Mean of the flux.**

**2. Standard Deviation of the flux.**

**3. Combined Differential Photometric Precision (CDPP).**

**Approach –**

* Lightcurve is an array of flux vs time. So we can find the mean of flux by using “mean()”
* Similarly we can get the standard deviation by using the “std()” function.
* Finally, we can estimate the CDPP value using “estimate\_cdpp()” function.

**Code** –

# 1. Mean of the flux

print("Mean : ", lc.flux.mean())

# 2. standard deviation of flux

print("Std : ", lc.flux.std())

# 3. CDPP metric

print("CDPP : ", lc.estimate\_cdpp())

**Output**  –

**Mean : 29228.791015625 electron / s**

**Std : 345.2061767578125 electron / s**

**CDPP : 1034.8816119957833 ppm**

**Observation** –

We can see that flux is rate of electron flow. Hence, we have electron/s unit in answer while CDPP is calculated in parts per million hence ppm unit.

**2. Transit Signal Detection and Analysis**

**(A) Plot and show the Box Least Square (BLS) periodogram of the modified light curve**

**Approach –**

* We will convert the lightcurve object to periodogram and choose the method as ‘BLS’
* We will plot the BLS plot which is plot of BLS power vs Period.

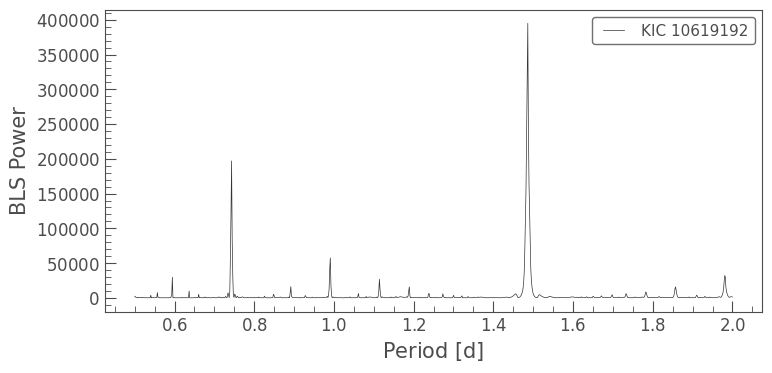
**Code** –

# converts the lightcurve to periodogram and plots it

period = np.linspace(0.5, 2, 1000)

bls = lc.to\_periodogram(method='bls', period=period)

bls.plot()



**(B) Using this BLS periodogram, find the following parameters –**

**1. Transit duration**

**2. The time period of revolution**

**Approach –**

* Once we have generated BLS object we can now access the information about various other parameters by calling respective functions.

**Code** –

# At maximum power we can find the transit duration and period after which it repeats

trans\_time = bls.transit\_time\_at\_max\_power

rev\_period = bls.period\_at\_max\_power

# 1. Transit duration

print("Transit duration : ", trans\_time)

# 2. Time period of revolution

print("Time period of revolution : ", rev\_period)

**Output**  –

**Transit duration : 908.3103621583249**

**Time period of revolution : 1.4864864864864864 d**

**(C) Verify the transit by phase folding modified light curve at the transit time and plot it.**

**Approach –**

* We will convert the period to phase using folding function available in lightkurve.
* We will plot the scatter plot and zoom a bit to locate the dip point of transit time.

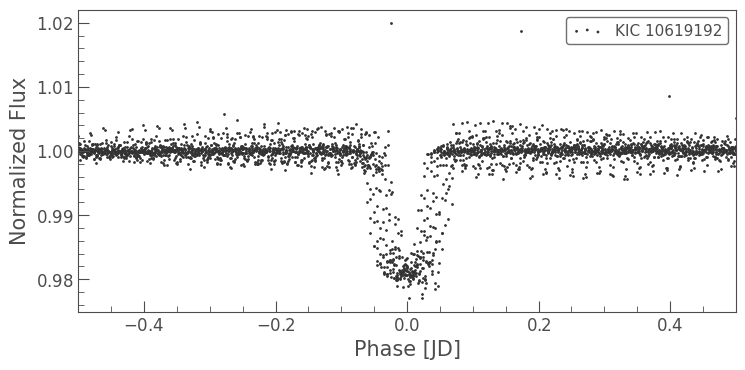
**Code** –

# fold the lightcurve plot at time = transit time

ax=lc.fold(period=rev\_period, epoch\_time=trans\_time).scatter()

# zooms in

ax.set\_xlim(-0.5, 0.5)



**Observation** –

We can see there is definitely a dip at the transit time. Which shows that our claim of exoplanet was true. There is change of flux observed due to intervention on planet between telescope and the star.

**(D) Make model curve of transit by plotting a best-fit curve through the folded data points**

**Approach –**

* We will first plot the folded curve which we obtained in the previous question.
* After that we will try to fit the best model curve which would be consisting of a simple rectangular dip at the transit time.

**Code** –

# Creates a BLS model using the BLS parameters

model = bls.get\_transit\_model(period=rev\_period, transit\_time = trans\_time, duration = planet\_b\_dur)

# we will combine this folded plot

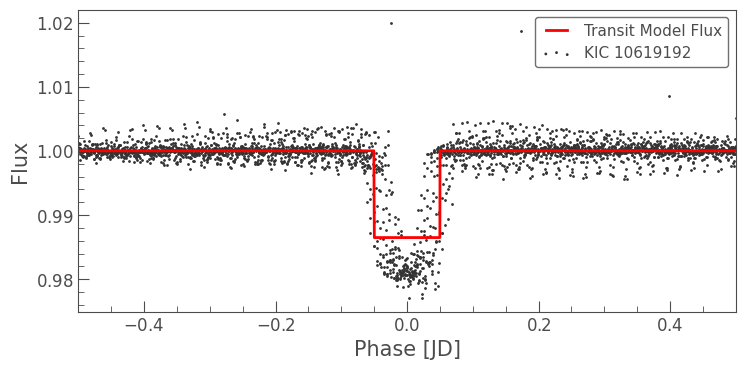
ax = lc.fold(rev\_period, trans\_time).scatter()

# then we will fit the best-fit curve through folded points

model.fold(rev\_period, trans\_time).plot(ax=ax, c='r', lw=2)

# zooming at dip location

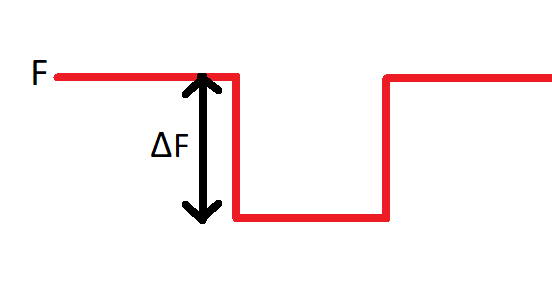
ax.set\_xlim(-0.5, 0.5)

****

**Explanation** –

By doing this we have now created a very simple model (drawn in red line) that replicates the behavior of our original BLS. This model is very useful to find the relative flux dip in next question.

**(E) Estimate the relative flux dip (ΔF/F)**

****

**Approach –**

* Above picture is the best-fit curve generated in previous question.
* We want the value of ΔF/Fwhich can be found be finding the value of F at dip region.
* Let say f = value of F at dip region, then ΔF = F – f
* Relative Flux dip = ΔF/F = (F – f )/F

**Code** –

# retuns the index of dip

model.fold(rev\_period, trans\_time).flux.argmin()

# prints 2087

# We will get the values of F and f by putting index as 0 and 2087

F = model.fold(rev\_period, trans\_time).flux[0]

f = model.fold(rev\_period, trans\_time).flux[2087]

d = F - f

print('Relative Flux Dip : ',d/F)

**Output**  –

**Relative flux Dip : 0.013538693326761216**

**3. Estimation of Stellar Parameters using Asteroseismology**

**(A) Search, download and plot short cadence light curves for quarters 2,5,6,7 of the star ‘Kepler-21’ (KIC = 3632418)**

**Approach –**

* First, we will iterate though the quarters 2,5,6,7 and search for the file with given KIC ID and short cadences.
* Then we will download their files and plot their lightcurves.

**Code** –

# iterates through all quarters

for i in [2,5,6,7]:

  # searches for files with KIC = 3632418 and short cadences

  search\_result = lk.search\_lightcurve('KIC 3632418',

                                       cadence='short',

                                       author='Kepler',

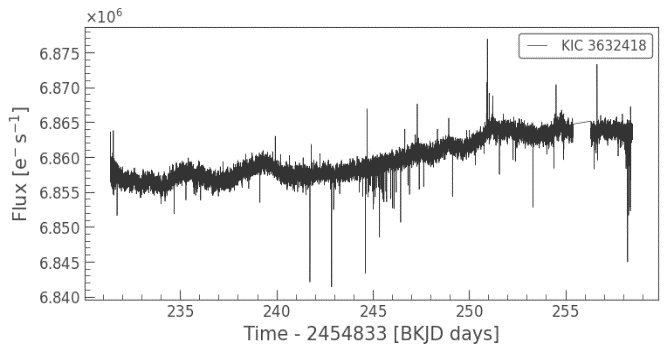
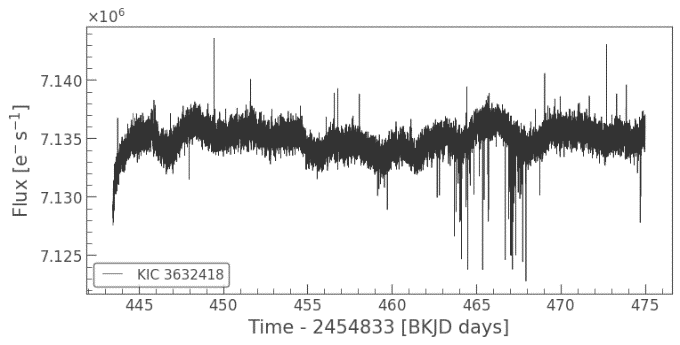
                                       quarter=i)

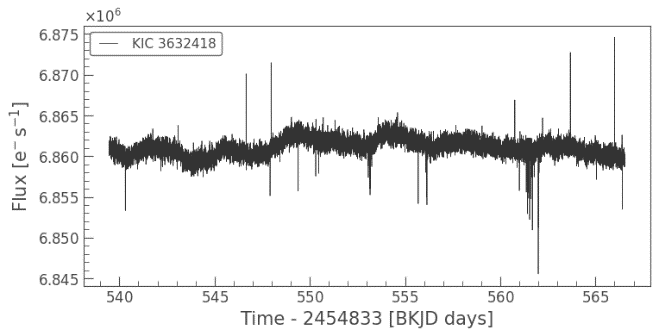
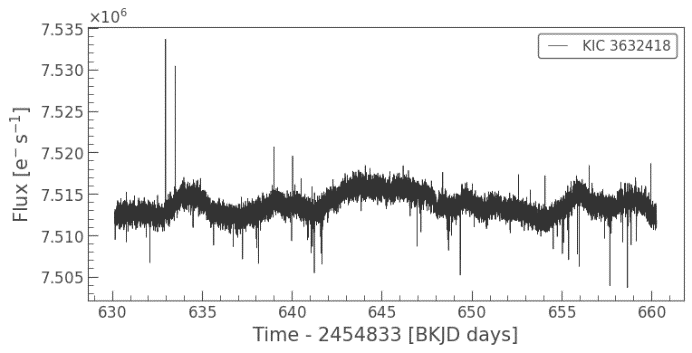
  # downloads the files as lightcurves

  lc = search\_result.download()

  # plots the lightcurves

  lc.plot()

**(B) Stitch, normalize and remove the long-term trends from the curve and plot it.**

**Approach –**

* As we have different offsets, we need to stitch all the lightcurves into one.
* After stitching we will normalize the lightcurve.
* We can remove the long term trends using Savitzky-Golay filter and plot it.

**Code** –

# searches all the files whose KIC=3632418 with short cadences and quarter = 2,5,6,7

search\_result = lk.search\_lightcurve("KIC 3632418",

                                     cadence='short',

                                     author='Kepler',

                                     quarter=(2,5,6,7))

# downloads all the files and stitches them into one lightcurve

lc = search\_result.download\_all().stitch()

# normalizes the lightcurve

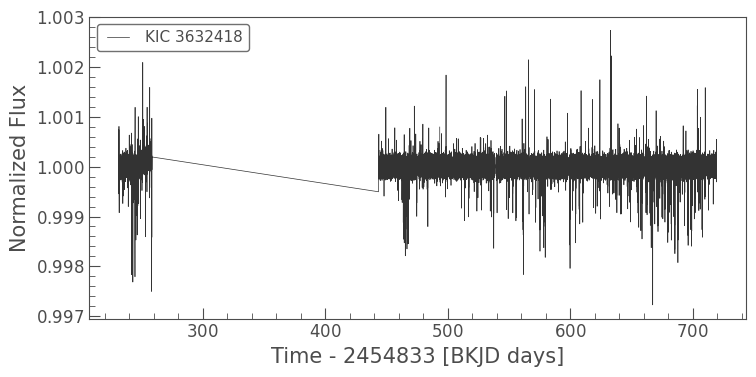
lc = lc.normalize()

# removes the long term trends using Savitzky-Golay filter

lc = lc.flatten()

# plot the modified plot

lc.plot()



**Observation –**

We can see that now our lightcurve looks flat and trendless after the above modifications.

**(C) Plot the periodogram**

**Approach –**

* We will convert the lightcurve into periodogram and plot it.
* We need to figure out the region where the peak occurs and so we can set the minimum and maximum frequency.

**Code** –

# convert the modified lightcurve to periodogram

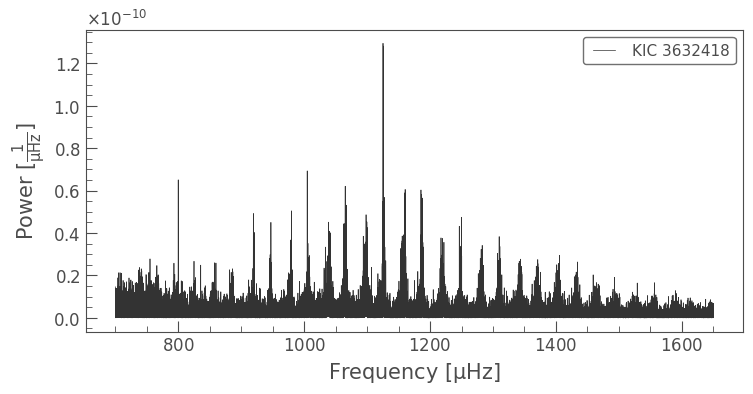
pg = lc.to\_periodogram(normalization='psd',

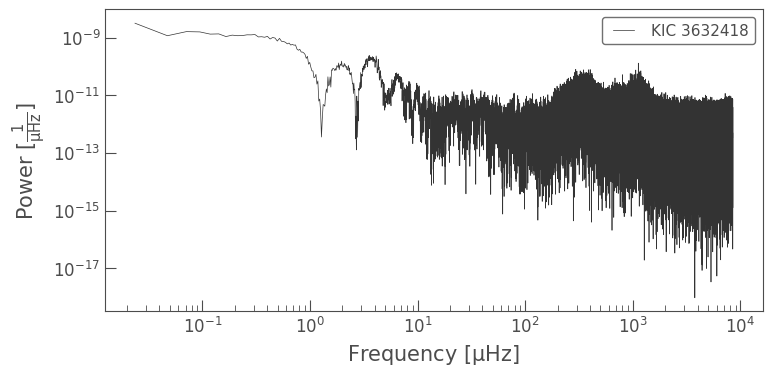
                       minimum\_frequency=700,

                       maximum\_frequency=1650)

# plots the periodogram

ax = pg.plot()



****

**Plot in “log scale” so as to cover the all frequency ranges and extreme values.**

**(D) Identify the region of Power excess power on the periodogram and plot it.**

**Approach –**

* From the previous plot we can see that max power occurs between 1000 to 1300.
* Upon this graph we can plot the smoothened graph obtained by “smooth” function.

**Code** –

# Plot a smoothed version of the power spectrum on top in green

ax = pg.plot(label='Original spectrum')

pg.smooth(filter\_width=1).plot(ax=ax,

                               color='green',

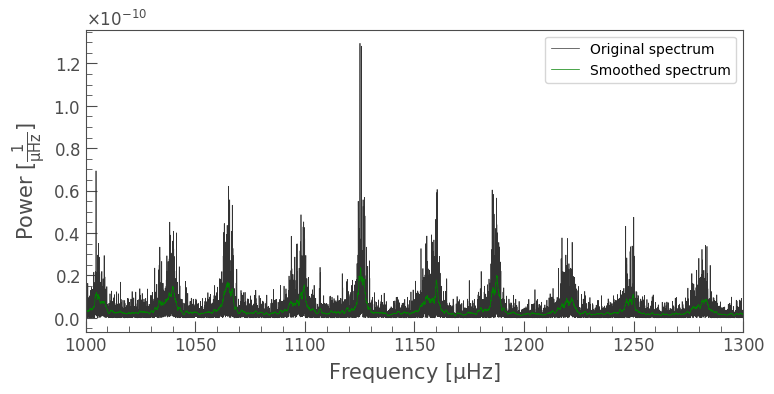
                               label='Smoothed spectrum')

# zoomes around the max power range

ax.set\_xlim(1000, 1300)

# plot the combined graphs

ax.legend()



**(E) Draw the best fit Gaussian curve enveloping the periodogram**

**Approach –**

* We will try to fit Gaussian equation on the periodogram plot by varying its parameters and tuning it till we get the satisfactory results.
* We can see that peak occurs at around 1125 uHz frequency so we can set the mean of Gaussian as 1125 and we will vary the standard deviation till it envelops the nearby spikes. Finally we will plot the gaussian curve on it.

**Code** –

# Plot a smoothed version of the power spectrum on top in green

ax = pg.plot(label='Original spectrum')

pg.smooth(filter\_width=1).plot(ax=ax,

                               color='green',

                               label='Smoothed spectrum')

# Highlight the "mode envelope" using a Gaussian curve

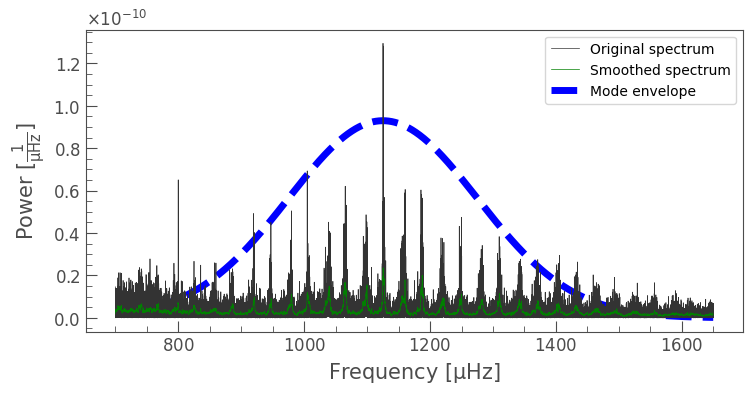
f = pg.frequency.value

ax.plot(f, 9.3e-11\*np.exp(-(f-1125)\*\*2/(2\*150\*\*2)),

        lw=5, ls='--', zorder=0,

        color='blue', label='Mode envelope');

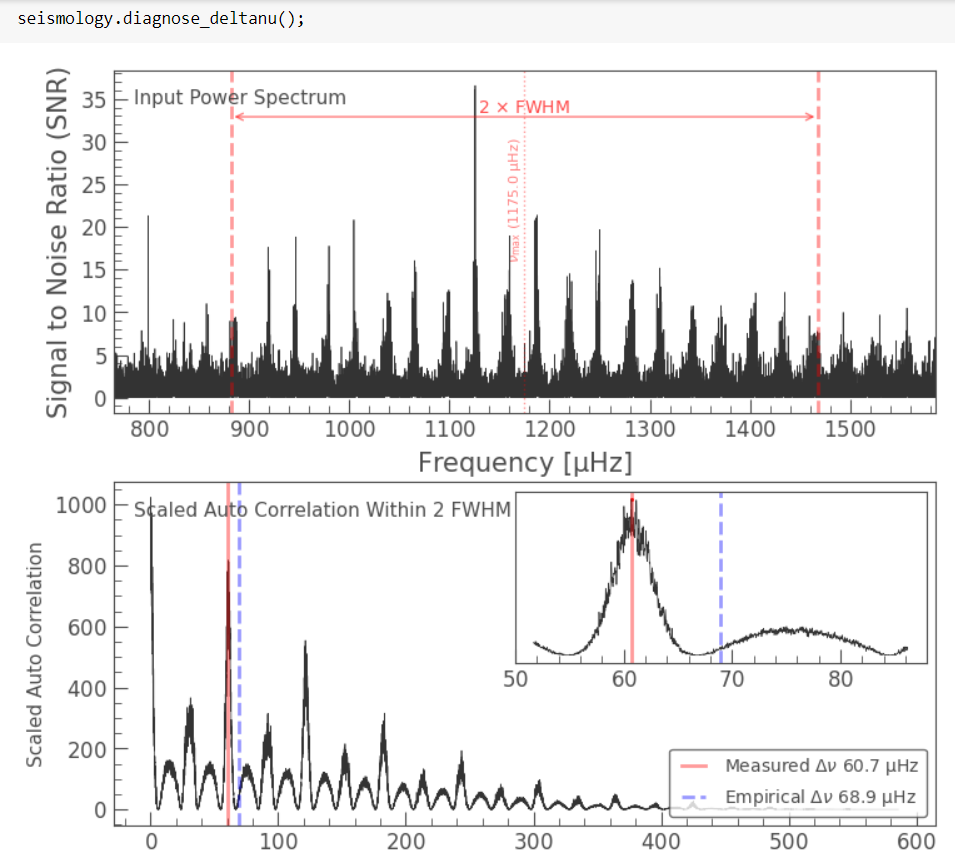
ax.legend()



**(F) From the Given periodogram find the numax and deltamax**

**Approach –**

* Numax: In the spectrum form above, we can see that the modes of oscillation rise up from the noise, reach a peak, and then decrease in power. This region, where the modes are visible, is called as *mode envelope*. The peak of this envelope is *frequency of maximum oscillation amplitude* or **“numax”**.
* Deltamax: A distinctive feature in the above spectrum is the equal spacing of the modes of oscillation. The spacing between two consecutive overtones of the same radial order called as *average large frequency spacing* called **“deltamax”**.
* For Numax: Now we will estimate the values of these metrics, using lightkurve tools. Before using Lightkurve tools, we will need to prepare our Periodogram object. We do this by first removing background noise using the flatten() method. To estimate numax, we used the 2D autocorrelation function (ACF) method. The Seismology object provides both estimate and diagnose methods. First, estimate\_numax() is called and this estimate\_numax()) function has measured a value from our SNR Periodogram, and has stored it as a property of the Seismology object.
* For Deltamax: In order to estimate Deltamax efficiently, we can use the fact that we only need to investigate the ACF in the region surrounding numax. we can infer the Full Width Half Maximum (FWHM) of the mode envelope based on the value of numax. This helps us further narrow down the region in which to calculate the ACF. So, first we used estimate\_deltanu(), and then diagnose\_deltanu() here and got the below observations and calculated deltamax.



**Code** –

# converts periodogram into signal to noise ratio

snr = pg.flatten()

# create seiosomlogic object from the signal to noise ratio plot

seismology = snr.to\_seismology()

# estimates numax

seismology.estimate\_numax()

# prints the numax value

print("numax : ",seismology.numax.value)

# estimates deltanu

seismology.estimate\_deltanu()

# prints deltanu

print("deltanu : ", seismology.deltanu.value)

**Output**  –

**Nu max : 1165.0**

**Delta nu : 60.72104606250446**

**(G) Find Mass and Radius of star**

**Approach –**

* From the seismologic object created previously we can print the mass and radius.

**Code** –

print("Mass : ", seismology.estimate\_mass())

print("Radius : ", seismology.estimate\_radius())

**Output**  –

**Mass : 1.4627661924732827 solMass**

**Radius : 1.9346498514008708 solRad**

**Observation** –

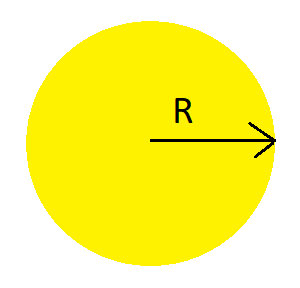
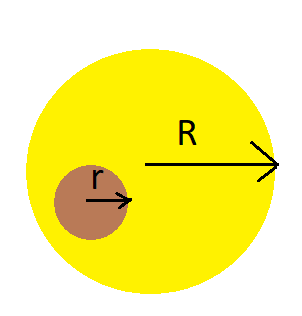
We can see that units are solar mass and solar radius so they are given with respect to sun’s mass and radius.

**4. Estimation of Planet Parameters from Transit Data**

**(A) Estimate Radius of the planet Kepler-17 b, if Radius of Kepler-17 is 1.01 solar radius**

**Approach –**

It is evident that the change in brightness of the star directly depends on the size of the planet as well as the star. Assuming that the stellar disc is of uniform brightness and neglecting any flux from the planet.

Flux ∝ Area Flux ∝ Area

Flux ∝ (π) Flux ∝ (π) – (π)

F1 = k(π) F2 = k[(π) – (π)]

Where k is proportionality constant.

Change in flux = ΔF = F1 – F2

= k(π)

The change in stellar flux to the absolute stellar flux can be expressed as

Relative Flux Dip = ΔF / F1

= k(π) / k(π)

= () / ()

We have already calculated Relative Flux dip in Question 2 and got as 0.0135387

So, substituting relative flux dip value,

**Radius of planet = 0.117519 solar radius**

**(B) Estimate the radius of the planet’s orbit if the stellar mass is 1.04 solar mass.**

**Approach –**

Assuming that the orbit is circular and applying Kepler’s third Law,

Where,

= 1.04 solar mass (Given)

Mass of Sun = 1.98847 kg

G = 6.67408

= 1.48649 days = 1.48649 X 86400 secs (from Question 2)

= radius of the planet’s orbit

= (

By substituting the value we get = 3813297.843 km

**Radius of orbit = 3813297.843 km**

**(C) Find out the mass of the planet if stellar velocity is 0.228 km/s**

**Approach –**

By Applying Angular Momentum Conservation law,

MSVS = MpVp

Given Vs = 0.228 km/s and Ms = 1.04 solar mass

From Question 2 b, To = 1.48649 days = 1.48649 X 86400 secs

From Above Question, Ro = 3813297.843 km

Vp = (2πRo)/To

Vp = 186.55 km/s

Putting in below equation,

MsVs = MpVp

Mp = (Ms\*Vs)/Vp

Mp = 1.271 \* solar mass

**Mass of planet = 1.271 \* solar mass**

Density = Mass / Volume

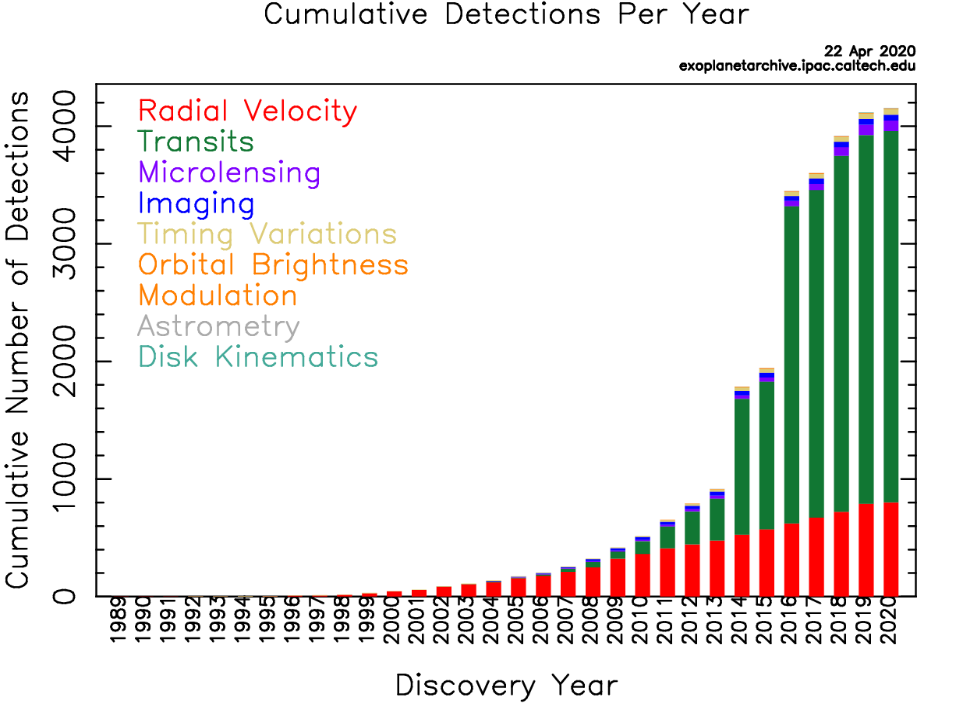
Substituting values of mass and radius from above, we get

**Density = 1104.39 kg /**

This density value falls in range of gases, hence this planet is **Gas Giant**

**Conclusion**

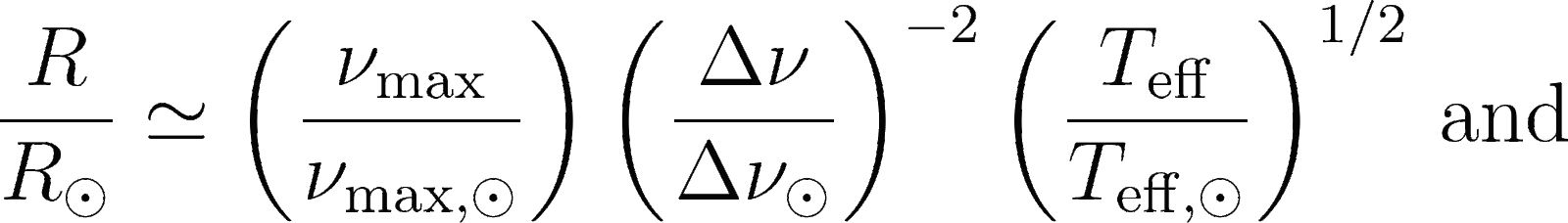
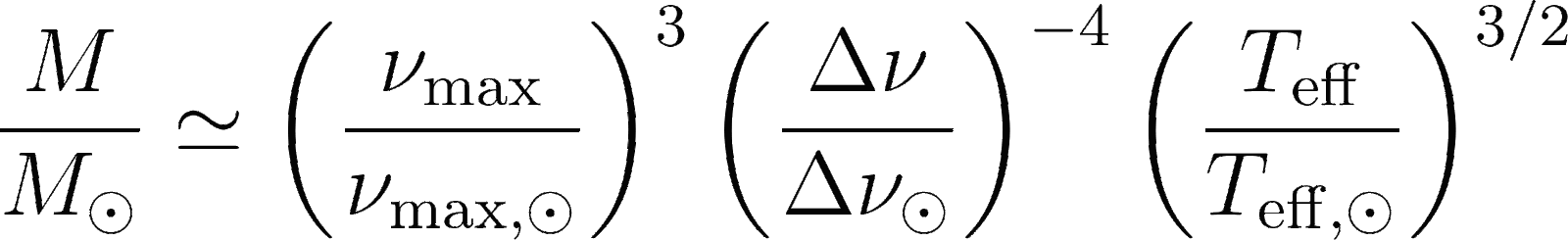
As we have seen from this Data analytics challenge that exoplanets detection using transits signal is very powerful tool from the following facts. Look at the number of exoplanets detected and what is the contribution of planets being detected by this method.

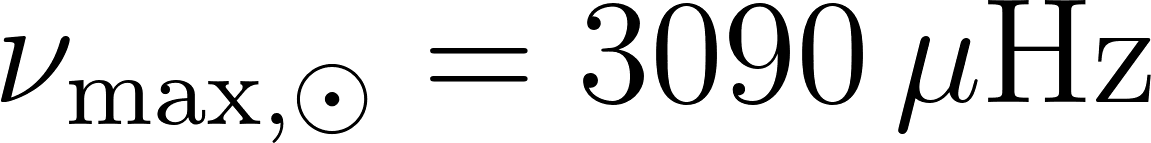


**Facts**

Till now, **4576** exoplanets have been detected and out of them **3445** have been detected by transits method. This method was first used in the year **1999** and the first planet detected was **hd209458b**

As we have seen, Asteroseismology is very powerful tool to analyse data of stars and planets and to add on that we can even find the temperature of star using the data of frequency from periodogram, mass and radius by following formula

[](https://latex-staging.easygenerator.com/eqneditor/editor.php?latex=%5Cfrac%7BR%7D%7BR_%5Codot%7D%20%5Csimeq%20%5Cleft(%5Cfrac%7B%5Cnu_%5Crm%7Bmax%7D%7D%7B%5Cnu_%5Crm%7Bmax%2C%5Codot%7D%7D%5Cright)%20%5Cleft(%5Cfrac%7B%5CDelta%5Cnu%7D%7B%5CDelta%5Cnu_%7B%5Codot%7D%7D%5Cright)%5E%7B-2%7D%5Cleft(%5Cfrac%7BT_%7B%5Crm%20eff%7D%7D%7BT_%7B%5Crm%20eff%2C%5Codot%7D%7D%5Cright)%5E%7B1%2F2%7D%5C%2C%20%5Ctext%7Band%7D#0) [](https://latex-staging.easygenerator.com/eqneditor/editor.php?latex=%20%5Cfrac%7BM%7D%7BM_%5Codot%7D%20%5Csimeq%20%5Cleft(%5Cfrac%7B%5Cnu_%5Crm%7Bmax%7D%7D%7B%5Cnu_%5Crm%7Bmax%2C%5Codot%7D%7D%5Cright)%5E3%20%5Cleft(%5Cfrac%7B%5CDelta%5Cnu%7D%7B%5CDelta%5Cnu_%7B%5Codot%7D%7D%5Cright)%5E%7B-4%7D%5Cleft(%5Cfrac%7BT_%7B%5Crm%20eff%7D%7D%7BT_%7B%5Crm%20eff%2C%5Codot%7D%7D%5Cright)%5E%7B3%2F2%7D%5C%2C%20#0)

Where [](https://www.codecogs.com/eqnedit.php?latex=%5Codot#0) symbol represents the Sun. The solar values used are   
[](https://www.codecogs.com/eqnedit.php?latex=%20%5Cnu_%7B%5Crm%20max%2C%20%5Codot%7D%20%3D%203090%5C%2C%20%5Cmu%5Crm%7BHz%7D#0) [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%5Cnu_%5Codot%20%3D%20135.1%5C%2C%20%5Cmu%7B%5Crm%20Hz%7D#0) [](https://www.codecogs.com/eqnedit.php?latex=T_%7B%5Crm%20eff%2C%20%5Codot%7D%20%3D%205777.2%5C%2C%20%5Crm%7BK%7D#0)

**References**

1. Identifying transiting exoplanet signals in light curve <http://docs.lightkurve.org/tutorials/3-science-examples/exoplanets-identifying-transiting-planet-signals.html>
2. Transit Photometry   
   <https://docs.google.com/document/d/1Un-1alVHShem0xTf0FpUS74alBQLg0Lc1pKCv3jKV9g/edit>
3. NASA Exoplanet Exploration  
   <https://exoplanets.nasa.gov/exoplanet-catalog/1815/kepler-17-b/>